Higher-Order Families

Vincent van Oostrom*
Institut für Informatik, Technische Universität München
80290 München, Germany
oostrom@informatik.tu-muenchen.de

Abstract

A redex *family* is a set of redexes which are 'created in the same way'. Families specify which redexes should be *shared* in any so-called *optimal* implementation of a rewriting system. We formalise the notion of family for orthogonal higher-order term rewriting systems (OHRSs). In order to comfort our formalisation of the intuitive concept of family, we actually provide three conceptually different formalisations, via *labelling*, *extraction* and *zigzag* and show them to be equivalent. This generalises the results known from literature and gives a firm theoretical basis for the optimal implementation of OHRSs.

1. Introduction

A computation of a result is *optimal* if its cost is minimal among all computations of the result. Taking rewrite steps as computational units the cost of a rewrite sequence is simply its length. Given a rewrite system the question then is: does an effective optimal strategy exist for it?

In the case of lambda calculus, a discouraging result was obtained in [BBKV76]: there does not exist a recursive optimal strategy. However, the observation made in [Lév78] was that it is possible to define an optimal strategy if one takes contraction of *families* of redexes as cost unit (the leftmost-outermost strategy works). Here the notion of family stands intuitively for a 'set of redexes which are created in the same way'.

Lévy provided three conceptually distinct formalisations (via *labelling*, *extraction* and *zigzag*) of the notion of family in the lambda calculus ([Lév78]), which he subsequently proved to be equivalent, thereby comforting the formalisation. Those results were later generalised to the class of *Interaction Systems* (ISs [AL94]). This generalisation required quite some effort and even failed in the case of zigzag.

We formalise all three characterisations of family for the class of orthogonal higher-order rewriting systems (OHRSs [Nipb]), in a natural way. In view of the above, this is a little surprising since the class of OHRSs properly contains the class of ISs. The main difference between our approach and the one in [AL94] is that we essentially put to use the decomposition of HRSs into a rule component and a substitution (simply typed λ-calculus) component ([Oos94, OR]). In a HRS rewrite step, the rule component takes care of the destruction and creation of function symbols and the substitution component takes care of the erasure and duplication of parts of the term.

Example 1. *Consider the term rewriting rule* $f(X,Y) \to g(Y,Y)$. *The step* $f(h,i) \to g(i,i)$ *using this rule* destroys f, creates g, erases h *and* duplicates i. *This becomes visible when we employ the decomposition mentioned above:* $f(h,i) \twoheadleftarrow_\beta (X.Y.f(X,Y))(h,i) \to (X.Y.g(Y,Y))(h,i) \twoheadrightarrow_\beta g(i,i)$. *Destruction of* f *and creation of* g *take part in the middle step, replacing the left-hand side of the rule by the right-hand side, and erasure of* h *and duplication of* i *are performed by the trailing* β*-steps.*

*Work partially performed at NTT BRL, Atsugi, Japan and VU, Amsterdam, The Netherlands.

The initial β-expansion is precisely there to make the subsequent HRS rule application possible. To simplify matters, we force this in general and such a β-expansion followed by a HRS rule application (but no β-reductions!) will be called a *partial* HRS step, denoted by $\rightharpoonup$.

Example 2 (Partial HRS-step). *The rewrite step in the previous example, can be decomposed as a partial HRS-step followed by a β-reduction:*

$$f(h,i) \rightharpoonup (X.Y.g(Y,Y))(h,i) \twoheadrightarrow_\beta g(i,i)$$

To clarify what we mean by 'precisely there' above, consider a step

$$(X.Y.f(X,Y))(h,i) \to (X.Y.g(Y,Y))(h,i)$$

This is not a partial HRS-step since its initial β-expansion (empty) does 'too little', i.e. it was not used to create (match) the X.Y.-part of the HRS rule application. Initial β-expansions should also not do 'too much' in a partial HRS-step. A sequence

$$f \twoheadleftarrow_\beta (x.x)(f) \to (x.x)(\ldots)$$

for a HRS rule $f \to \ldots$ is not a partial HRS-step, since the initial expansion was useless for performing the step.

As we see it the problems with the formalisation of the family relation reported in [AL94] are due to the fact that in a single HRS step symbols (redexes) can be first created and then erased or duplicated. This cannot happen in either (first-order) TRSs or lambda calculus. In the rest of the paper, we use the following example to illustrate our methods.

Example 3 (Running). *1. Consider the OHRS:*

$$\begin{array}{rcl} f & \to_f & \ldots \\ g(x.X(x)) & \to_\delta & X(f) \end{array}$$

It yields a rewrite step $\sigma: g(x.h(x,x)) \to_\delta h(f,f)$ in which f is first created and then duplicated in the same step, hence intuitively the two occurrences of f in the final term belong to the same family. This becomes visible in its decomposition:

$$\tilde{\sigma}: g(x.h(x,x)) \rightharpoonup (X.X(f))(x.h(x,x)) \twoheadrightarrow_\beta h(f,f)$$

The partial HRS-step creates f, after which it is duplicated in the subsequent β-reduction.

2. *Consider the OHRS:*

$$\begin{array}{rcl} f & \to_f & \ldots \\ g(X) & \to_\epsilon & f \\ \mu(x.X(x)) & \to_\mu & X(\mu(x.X(x))) \end{array}$$

and the rewrite sequences

$$\tau: M = \mu(x.g(x)) \to_\mu g(\mu(x.g(x))) \to_\epsilon g(\mu(x.f)) = N$$

$$\upsilon: M = \mu(x.g(x)) \to_\epsilon \mu(x.f) = O$$

Intuitively, the two occurences of the redex f in the final terms of these sequences are in the same family. This becomes visible in their decompositions

$$\tilde{\tau}: M \rightharpoonup_\mu (X.X(\mu(x.X(x))))(x.g(x)) \twoheadrightarrow_\beta g(\mu(x.g(x))) \rightharpoonup_\epsilon g(\mu(x.(X.f)(x))) \twoheadrightarrow_\beta N$$

and

$$\tilde{\upsilon}: M \rightharpoonup_\epsilon \mu(x.(X.f)(x)) \twoheadrightarrow_\beta O$$

In both cases only g is 'responsible for creating' f.

3. *To disable trivial solutions to the formalisation of the family relation, we will also consider the following non-example. Consider the rule:*

$$f \quad \to_f \quad f$$

the redex f in the initial term of the step ϕ: $f \to_f f$ is intuitively not in the same family as the redex f in its final term, since their 'creation histories' differ.

To show equivalence of the three approaches to the family relation, we work directly with the decomposition as in the example such that destruction and creation are caused by partial HRS-steps and erasure and duplication are caused by β-steps.

Some familiarity with term rewriting ([DJ]), the lambda calculus ([Bar84]), higher-order term rewriting ([Nipb]), and the study of optimality for those classes of rewriting systems ([Mar92, Lév78, AL94]) is assumed. Since we need to cover quite some (technical) ground, we will mostly be satisfied with giving proofsketches and clarifying examples.

We briefly recapitulate some general no(ta)tions from rewriting theory after which we present higher-order term rewriting systems from the point of view of structured rewriting systems.

We employ common relational notation, e.g. for R a relation R^{-1}, $\complement R$, $R^{=}$, R^{+}, and R^{*} denote its *inverse*, *negation*, *reflexive closure*, *transitive closure*, and *reflexive-transitive closure* respecitvely. Let S be another relation. Then $R\,;S$, $R \cap S$, and $R \cup S$ denote their *sequential compostion*, *intersection*, and *union*. An *order* ($\succ$, $\sqsupset$) is a transitive relation. It is *quasi* when reflexive, *partial* when anti-symmetric, *strict* when irreflexive, an *equivalence* when symmetric and a *well-founded* order when there is no infinite strict sequence $a \succ b \succ \ldots$.

Definition 1 (Abstract Rewriting System (ARS)). *An* abstract rewriting system *(ARS) is a binary relation on a set. We use arrows to denote ARSs which allows for some special rewriting notation. normal form. Let $\to$ be an ARS, then $\leftarrow =^{\mathrm{def}} \to^{-1}$, $\twoheadrightarrow =^{\mathrm{def}} \to^{*}$, $\to^{!}$ is $\twoheadrightarrow$ with second component a normal form, $\leftrightarrow =^{\mathrm{def}} \leftarrow \cup \to$ (*symmetric closure*). A rewrite sequence σ, τ, υ is a sequence of rewrite steps, such that each step start at the object its predecessor ends in. The length $|\sigma|$ of a rewrite sequence is its number of steps.*

2. Higher-Order Rewriting Systems

Properties of rewriting systems can rougly be divided into combinatorial (e.g. strong normalisation) and structural ones (e.g. orthogonality). *Structured rewriting systems* make this division explicit by having a combinatorial component (rules) and a structural component (substitutions).[1] Structured rewriting can then be viewed as 'rules modulo substitutions'.

Definition 2 (Structured Rewriting System (SRS)). *1. Let $\mathcal{SC}$ be a* Substitution Calculus (SC) *consisting of:*

a set $A, B, C \in \mathcal{T}$ of types,

a set $M, N, O \in \mathbb{PT}$ of prestructures *inhabiting types in $\mathcal{T}$. We assume for every type A infinitely many atomic prestructures x^A, y^A, X^A, Y^A, …* variables *of that type,*

an ARS $\to_{\mathcal{SC}}$ on prestructures.

Among the variables a subset $f, g, h \in \mathcal{A}$ of symbols *is distinguished. A* structure *is a prestructure in $\to_{\mathcal{SC}}$-normal form. A* rewrite rule *$\aleph$ is a triple $\langle l, r, A\rangle$, often simply denoted by $l \to r$, of closed (may contain symbols in $\mathcal{A}$ but no other (free) variables) structures l and r proving the same formula A. The ARS $\to_\aleph$ on structures is defined by $M \to_\aleph N$, if $M \leftrightarrow^{*}_{\mathcal{SC}} C\boxed{l}$ and $C\boxed{r} \leftrightarrow^{*}_{\mathcal{SC}} N$, where C is a* context, *i.e. an $\to_{\mathcal{SC}}$-normal form containing one occurrence of a variable $\Box^A$. The rewrite step is said to* contract *the* ($\aleph$)-redex (C,$\aleph$).

[1] The substitution component can be a structured rewriting systems itself again, &c.

2. *A* Structured Rewriting System $\mathcal{H}$ *is a triple* $\langle \mathcal{SC}, \mathcal{A}, \mathcal{R} \rangle$ *consisting of a substitution calculus* $\mathcal{SC}$, *an alphabet* $\mathcal{A}$ *and a set* $\mathcal{R}$ *of rewrite rules. The ARS associated to* $\mathcal{H}$ *is a relation on* $\mathbb{PT}$ *defined by* $\rightarrow_{\mathcal{H}} =^{\mathrm{def}} \bigcup \aleph \in \mathcal{R}. \rightarrow_{\aleph}$.

In this paper we consider structured rewriting modulo simply typed λ-calculus, i.e. higher-order term rewriting. As a consequence of confluence of λ-calculus each rewrite step can be written as an *expansion* (unplugging, matching) $M \twoheadleftarrow_{\beta\overline{\eta}} C\boxed{l}$, followed by a *replacement* $C\boxed{l} \rightarrow_{C\boxed{l \rightarrow r}} C\boxed{r}$, and a *reduction* (plugging, normalisation) $C\boxed{r} \twoheadrightarrow_{\beta\overline{\eta}} N$. The replacement is said to *destroy* l and to *create* r. In the normalisation phase *erasure* or *duplication* of parts of C can occur (see Example 1). Destruction and creation are caused by the rule component while erasure and duplication are caused by the substitution component of a structured rewriting system.

Definition 3 (Higher-Order Term Rewriting Systems). *1. Simple types (propositional formulae) are specified by the grammar* $A ::= o \mid A \rightarrow A$. *A* preterm[2] *is an object* M *such that* $M{:}A$ *for some type* A *can be inferred from*

(var) $x^A{:}A$ *for termvariables* x,

(app) $M{:}A \rightarrow B,\ N{:}A \Longrightarrow M(N){:}B$.

(abs) $M{:}B \Longrightarrow x^A.M{:}A \rightarrow B$.

The rewrite relation $\rightarrow_{\mathcal{SC}}$ *on preterms modulo the usual renaming of bound variables* (α-*conversion, see [Bar84]), is generated by the rules*

$$\begin{array}{rcl} (x.M)(N) & \rightarrow_\beta & M^{[x \mapsto N]} \\ M & \rightarrow_{\overline{\eta}} & x.M(x) \end{array}$$

where $\overline{\eta}$ *is not allowed to create* β-*steps and* x *does not occur in* M.

2. *A* Higher-Order Term Rewriting System *(HOTRS [Wol93]) is a structured rewriting system having simply typed* λ-*calculus as substitution calculus. To stress the fact that symbols of an HOTRS correspond to usual* function *symbols, we employ the notation* $f(M_1, \ldots, M_m)$ *instead of* $f(M_1)\ldots(M_m)$ *in case* $m > 1$ *(cf. [Nipa]).*

3. *A* pattern *([Mil]) is a term of the form* $\vec{X}.M$ *such that*

   (a) $M{:}o$,

   (b) M *is of the form* $f(M_1, \ldots, M_m)$, f *is called the* head-symbol *of the term,*

   (c) each occurrence of an X_k *among* $\vec{X}$ *in* M *only has (*$\overline{\eta}$*-normal forms of) pairwise distinct variables not among* $\vec{X}$ *as arguments.*

The initial sequence $\vec{X}$ *is called the* binder, *the variables (not) in it (*bound*)* free *variables, and an occurrence* $X(\vec{M})$ *in* r *(so* $\vec{M}$ *is a sequence of* η*-expanded bound variables) is called a* binding hole *of the pattern (rule). A* Higher-order Rewriting System *(HRS [Nipa]) is a HOTRS where all left-hand sides of rules are patterns. We use* $\mathcal{H}$ *to denote HRSs.*

We often omit the binder of a HRS rule and write $l \rightarrow r$ instead of $\vec{X}.l \rightarrow \vec{X}.r$. We notationally distinguish free and bound variables by using capitals (X, Y) for the former and small letters (x, y) for the latter.

Patterns can be viewed as the higher-order formalisation of the intuitive notion of 'contiguous substructure'. Defining the *encompassment* relation $\bullet\!\trianglerighteq$ by $C\boxed{M} \bullet\!\trianglerighteq M$ if C is a context and M a pattern, it is clearly a generalisation of the first-order encompassment ordering (see [DJ]) and it inherits from the first order case nice properties such as decidability of and closure under unification ([Mil]). Note however that we have $f(x.X(x)) \bullet\!\trianglerighteq f(x.Y)$ but not vice-versa due to the fact that

[2]Preterms are just simply typed λ-terms. We use functional notation and write $x.M$ instead of the usual $\lambda x.M$ in order to stress that the preterms are merely used as a notational system.

one cannot substitute a term containing the bound variable x for Y. This means that HRS rules can perform a 'no occur' check for a bound variable (e.g. the eta rule in lambda calculus). This is a nonlocal check and causes the theory of permutation equivalence (see Definition 11) to break down. For that reason we make the following assumption.

> Every free variable in the left-hand side of a rule has all bound variables in which scope it is, as arguments.

Such left-hand sides are called *fully extended* in [HP95].

For our decomposition as exemplified in the introduction we need to rewrite not only on terms, but also on preterms. This is done by dropping some restrictions on the context $C\Box$ in a replacement step $C\boxed{l} \to_{C\boxed{l\to r}} C\boxed{r}$. We do not require it to be in β-normal form any more and neither require that it contains only one occurrence of the hole. We still do require that the context is in $\overline{\eta}$-normal form and the hole should of course be of the right type.

Note that we may assume wlog that all substitution calculus steps are (ordinary or inverse) β-steps (cf. [Oos94, Pro. 3.2.10]). We even may wlog η-normalise the bound holes in left-hand sides of rules (i.e. bound holes then look like $X(\vec{x})$ for bound variables $\vec{x}$ instead of their $\overline{\eta}$-normal form). Assuming all this, we define the decomposition.

Definition 4 (Partial HRS-step). *Let M, N be preterms. A* partial *HRS-step $M \rightharpoonup_{C\boxed{l}r} N$ can be decomposed as a sequence $M \twoheadleftarrow_{\sigma} C\boxed{l} \to_{C\boxed{l\to r}} C\boxed{r} = N$, where σ^{-1} is a maximal standard β-rewrite sequence in which in each step a descendant of a symbol (abstraction or application) in l is destroyed (with the usual descendant relation for λ-calculi, see Figure 1).*

The β-expansion steps in a partial HRS-step are precisely those used to 'split off' the left-hand side from its surroundings. Its length is completely fixed by the shape of the left-hand side; actually it's the number of (free and bound) variables occurring in its bounding holes. Furtermore, although the left-hand side of the applied rule is not (necessarily) in $\overline{\eta}$-normal form, both the context $C\Box$ and the right-hand side r are, yielding that partial HRS-steps preserve $\overline{\eta}$-normal forms. See Example 2 above for some (non-)examples.

A redex family consists of a set of redexes in a structure which are created 'in the same way'. The contraction of a family of redexes is the simultaneous contraction of all the redexes in the family. This simultaneous contraction makes sense if contracting a redex is a *local* action and if the redexes in a family are *far enough apart*, such that they operate on disjoint parts of the structure. We recapitulate the main results of *orthogonal* rewriting and refer to [Nipb, Oos94] for the details.

Definition 5 (Orthogonality). *A HRS is* orthogonal *if it is*

1. left-linear*, that is, each free variable in a left-hand side of a rule occurs only once.*
2. non-ambiguous*, that is, left-hand sides of rules cannot be unified (except for the trivial unification of a left-hand side of a rule with itself at the root).*

Contraction of a set of redexes in an orthogonal HRS is called a *complete development* step of that set.

Theorem 1 (Finite Developments (FD)). *All serialisations (splitting into single HRS steps) of complete development-steps are finite and end in the same term.*

We will often confuse developments with their serialisations in the sequel.

3. Optimality

In literature ([Lév78, AL94]), a family relation is an equivalence relation on pairs consisting of a rewrite sequence and a redex in the final term of the sequence. We deviate from this in two ways. The first and main deviation will be that for the first component we do not consider HRS rewrite sequences, but work on their decomposition as rewrite sequences consisting of partial HRS- and β-steps, The second modification is that the second component can be any part of a λ-term not just a redex.

Definition 6 (Family Relation). *A* locked *sequence is a structure $\sigma \lhd \partial$, where $\sigma : M \twoheadrightarrow N$ is a sequence (on preterms) and ∂ a* part *of N, meaning that ∂ is a pair consisting of a position ϕ and a λ-term O, such that $N = C[O^\theta]_\phi$ for some λ-calcus context C and substitution θ. The sequence is said to be* locked on to O (at ϕ). *When clear from the context, we often omit explicit mention of ϕ and confuse a lock with its λ-term. A* family *relation $\simeq$ is an equivalence relation on coinitial rewrite sequences locked on to the same lock (i.e. their λ-terms are identical).*

The following definition of optimality is parametrised over the notion of family.

Definition 7 (Family Rewrite). *A* family *rewrite is a $\multimap\!\!\rightarrow$-rewrite in which in each step a non-empty* family *of redexes is contracted. A family rewrite σ to normal form is* optimal *if $|\sigma| \leqslant |\tau|$ for all $\multimap\!\!\rightarrow$-rewrites τ starting and ending in the same preterms as σ. A rewrite strategy is optimal if it constructs optimal family rewrites.*

Our (re)main(ing) task will be to formalise the three different concepts of family (via *labelling*, *extraction* and *zigzag*) provided for the lambda calculus in [Lév78], but now in the case of orthogonal HRSs.

4. Forth

The purpose of *labelling* is to keep track of the creation history of a structure through time (rewriting), in the structure itself. This is done by equiping the elements of a structure with a label to record their history. Applying a rewrite step then consists of replacing a substructure together with its history by a new substructure whose history should record the old one and additionaly how it has been created. Substructures having the same history are declared to be in the same *forward* family.

In first-order TRSs and λ-calculus this idea is still pretty easy to implement (see e.g. [Mar92, Lév78]), but the situation becomes quickly more complicated for higher-order systems (cf. [Klo80, AL94] for the second order case).

Our solution is to first conceive of labellings for HRS rules and the substitution calculus separately and then to combine them. The idea for labelling HRS rules is that the created elements get as label the labels of the destroyed elements together with the rule, and a sequence number (distinct for distinct elements of the right-hand side). Moreover, the edges connected to splitt-off parts get a number indicating where they were connected to the left-hand side. For the λ-calculus the labelling introduced in [Lév78] is employed.

Example 4 (See Example 3). *In this example, we have simplified the labelling in order to improve (still terrible) readability. We forget about the labelling of the substitution calculus and since at most one function symbol is created by the rules in the examples, we forget about the sequence number in the labels. β-steps do not change the labelling of function symbols and initially, distinct (occurrences of) function symbols have distinct labels (the labelling is* initial*).*

1. *Labelling $\tilde{\sigma}$ yields*

$$g^a(x.h^b(x,x)) \rightharpoonup_\delta (X.X(f^{(a,\delta)}))(x.h^b(x,x)) \twoheadrightarrow_\beta h^b(f^{(a,\delta)}, f^{(a,\delta)})$$

and we see that both occurrences of f have label (a,δ) so are in the same forward family.

2. *Labelling the first step of the (decomposition) of τ yields*

$$\mu^a(x.g^b(x)) \rightharpoonup_\mu (X.X(\mu^{(a,\mu)}(x.X(x))))(x.g^b(x)) \twoheadrightarrow_\beta g^b(\mu^{(a,\mu)}(x.g^b(x)))$$

and the second step becomes

$$g^b(\mu^{(a,\mu)}(x.g^b(x))) \rightharpoonup_\epsilon g^b(\mu^{(a,\mu)}(x.(X.f^{(b,g)})(x))) \twoheadrightarrow_\beta g^b(\mu^{(a,\mu)}(x.f^{(b,g)}))$$

3. *Labelling $\tilde{\upsilon}$ gives*

$$\mu^a(x.g^b(x)) \rightharpoonup_\epsilon \mu^a(x.(X.f^{(b,g)})(x)) \twoheadrightarrow_\beta \mu^a(x.f^{(b,g)})$$

Both occurrences of the symbol f in the final terms of the labellings of $\tilde{\tau}$ and $\tilde{\upsilon}$ have the label (b,g), so they are in the same forward family.

4. *Labelling ϕ yields $f^a \rightarrow_f f^{(a,f)}$ and clearly the occurrences of f in the initial and final term have distinct labels (a and (a,f)).*

Definition 8 (Labelled HRS). 1. *Let $(a, b \in)$ $\mathcal{L}$ be a set of labels. The following grammar defines the set of* λ-labels.

$$\alpha \quad ::= \quad a \mid \alpha \,;\, \alpha$$

where ; denotes string composition. The labelled λ-calculus λ^ℓ is obtained for the unlabelled λ by labelling elements (nodes and edges) of the (Bourbaki) graph representation. Since this graph representation is relatively standard by now (see e.g. [Lan93]) we refrain from presenting the (tedious) formal definition and show a picture instead (Figure 1).

2. *Steps in a labelled HRS are either*

   (a) *labelled β-steps as defined above, or*

   (b) *labelled partial HRS-steps. Elements of the set $\mathcal{L}$ of* rule-labels *are defined by the grammar:*

$$a \quad ::= \quad \langle \vec{\alpha}, l \rightarrow r, m \rangle \mid m$$

where m ranges over natural numbers. For the construction of the labelled partial HRS-step, first perform the labelled β-expansion as defined above (this only depends on the left-hand side). Then perform the (unlabelled) HRS replacement step and after that label all created (either by the β-expansion or by the replacement) elements by $\langle \vec{\alpha}, l \rightarrow r, m \rangle$, where α is an enumeration (fixed by l) of the labels in the left-hand side, and m is a sequence number in an enumeration of the elements (fixed by r and the β-expansion). Finally, the edges connecting the created symbols to its surroundings are provided with a sequence number (establishing the interface).

We will use I, J to denote labelling functions. The labelled preterm obtained by labelling the preterm M using I is denoted by M^I. There is a tight correspondence between an orthogonal HRS and its labelled version.

Proposition 1. *Consider an orthogonal HRS and its labelled version.*

1. *Let M^I be a labelling of M. Every partial HRS-step (β-step) $M \rightarrow N$ can be lifted in a canonical way to a labelled HRS step (β-step) $M^I \rightarrow N^J$ for some labelling J of N.*

2. *A labelled partial HRS-step (β-step) $M^I \rightarrow N^J$ projects in a canonical way onto a partial HRS-step (β-step) $M \rightarrow N$.*

Proof

1. For β-steps the lifting property is well-known (see e.g. [Lév78]). For partial HRS-steps, it suffices to check that the construction is deterministic.

2. Forgetting the labelling in the decomposition of a labelled HRS step yields the decomposition of a HRS step. □

Definition 9 (Initial Labelling). *In an* initial *labelling of a preterm all elements of the labelled preterm have distinct atomic (not composed of other) labels.*

By the proposition labellings of terms can be uniquely extended to labellings of rewrite sequences via the labelling of their initial term. In particular, initially labelling the initial term of a (locked) rewrite sequence yields a unique (locked) labelled rewrite sequence, called its *initial* labelling. The labelling of a part ∂ of a term, is called the *degree* of that part.

Definition 10 (Forward Family). *Let $\sigma \lhd \partial$, $\tau \lhd \partial$ be two locked rewrite sequences. They are in the same* forward family, *$\sigma \lhd \partial \simeq_f \tau \lhd \partial$, if their initial labellings are locked onto the same labelled part.*

Since labelled term equality is an equivalence relation, the forward family relation is an equivalence relation as well.

5. Back

A characterisation of families which is at first sight unrelated to the one via labelling works via the so-called extraction procedure. Rewrite sequences are first sorted in an outside-in order after which exactly those steps which contribute to the redex in the final term are extracted from the rewrite sequence. The former procedure is known as *standardisation* and the latter as *extraction*. Rewrite sequences which extract into the same sequence after standardisation are declared to be in the same *backward* family.

Example 5 (See Example 3). *The sequences we consider in this example are already in outside-in order, so to show that a redex in the final terms of two rewrite sequences are in the same backward family, we only need to show that they extract into the same sequence.*

1. *In $\tilde{\sigma}$ only the initial partial HRS-step contributes to the creation of either occurrence of f, so this sequence extracts in both cases into*

$$\ddot{\sigma}\colon g(x.h(x,x)) \rightharpoonup_\delta (X.X(f))(x.h(x,x))$$

with the same (unique) occurrence of the redex f in the final term, showing that both f-redexes belong to the same backward family. Note that the resulting rewrite sequence is not a HRS rewrite sequence, since the final term is not in β-normal form; it is only a partial HRS step.

2. *The first HRS step in $\tilde{\tau}$ does not contribute to f at all, so it can be 'thrown away', resulting in $\tilde{\upsilon}$ and we can already conclude from this (and the fact that the extraction procedure gives a unique result) that the occurrences of f in the final terms of $\tilde{\tau}$ and $\tilde{\upsilon}$ are in the same backward family.*

3. *As in the first item, $\tilde{\upsilon}$ extracts to a partial HRS step $\ddot{\upsilon}$ since the final erasing β-step does not contribute to f at all,*

$$\ddot{\upsilon}\colon \mu(x.g(x)) \rightharpoonup_\epsilon \mu(x.(X.f)(x))$$

4. *Both the redex (empty sequence) f and the sequence $f \to_f f$ are in extraction normal form, so the redex f in the initial term of the step is not in the same backward family as f in its final term.*

If HRS steps are considered (not decomposed ones), then in the first item it is not possible to relate the two redexes to each other as was noted in [AL94, Exa. 7] (note that in order to be related both the extracted rewrite sequences *and* the redex occurrences must be the same), due to the fact that σ is a 'create and duplicate' HRS step.

5.1. Standardisation

Adopting development steps as basic steps induces a *permutation equivalence* relation on sequences consisting of single steps in much the same way as true concurrency induces interleaving concurrency (cf. [Lan93]).

Definition 11 (Permutation). *An* elementary diagram *(ED) is a pair* (σ,τ)*, where both* sides $\sigma = u \,;\, \sigma'$ *and* $\tau = v \,;\, \tau'$ *are complete developments of* $\{u,v\}$*. If* $u = v$ *the diagram is* trivial*. The* permutation *relation* $\equiv^1$ *between rewrite sequences is defined by* $v \,;\, \sigma \,;\, \phi \equiv^1 v \,;\, \tau \,;\, \phi$ *if* (σ,τ) *is an elementary diagram.* Permutation equivalence $\equiv$ *is defined as the equivalence closure of* $\equiv^1$.

Permutations only change the order in which, but not how structures are created and this motivates defining backward families to be invariant with respect to permutation equivalence. Then, in order for the family relation to be effective the permutation equivalence relation has to be effective. One way of achieving this is to give an effective procedure for constructing unique representatives of $\equiv$-equivalence classes. *Standardisation* sorts the steps first into an outside-in order and then into a left-to-right order. The resulting sorted sequence is called *standard*. Just like comparison-based sorting procedures work by removing 'inversions', standardisation procedures work by removing 'anti-standard' pairs, where a sequence contains an *anti-standard* pair if sometime a redex is contracted and later a redex outside it to which it didn't contribute is (cf. [Klo80, Def. I.10.2.1]).

The standardisation method we present extends (and corrects) the one of [Klo80, Sec. II.6] to orthogonal HRSs. The method is based on a kind of worst-case analysis showing that repeatedly removing an arbitrary anti-standard pair terminates and yields a unique standard sequence. In literature one can find 'efficient' standardisation prcoedures, e.g. the (non-deterministic) algorithm **STD** in [GLM] roughly corresponds to bubblesort.

Definition 12 (Standardisation). *1. Rewrite steps (either* β*- or partial HRS-steps) are ordered via the positions of their constitutent elements. In particular* $u \leq v$ *for steps* u*,* v *if there is a position affected by* u *which is a prefix of a position affected by* v*. Derived relations are* $\geq =^{\mathrm{def}} \leq^{-1}$*,* $< =^{\mathrm{def}} \leq - \geq$ *(*outside*),* $> =^{\mathrm{def}} <^{-1}$ *(*inside*),* $\mid =^{\mathrm{def}} \complement\!\leq \cap \complement\!\geq$ *(*parallel*),* $\asymp =^{\mathrm{def}} \leq \cap \geq$ *(*overlap*). The left-side of an elementary diagram is related to the right-side via the relationship between their initial steps and this is naturally extended to permutations. The relations on redexes are transitive (for orthogonal systems), but this does not hold anymore for their extensions to rewrite sequences. For that reason, we define* $\| =^{\mathrm{def}} |^*$ *on rewrite sequences.*

2. *Let* $\Rrightarrow =^{\mathrm{def}} \| \,;\, > \,;\, \|$ *be the* standardisation *relation on rewrite sequences. A rewrite sequence is said to be* standard *if it is in* $\Rrightarrow$*-normal form. Standardisation of a locked sequence is standardisation of its rewrite sequence component, leaving the lock untouched.*

The permutation relation $\equiv^1$ is partitioned into $<$, $>$, $\|$, and $\asymp$. Note that since partial HRS-steps and β-steps do not have overlap with each other or themselves by orthogonality, $\asymp$ is the identity on these steps, yielding that $\leq$ is a partial order on them.

Example 6. *1. Consider a HRS having three rules, with respective left-hand sides:* $f(X,Y)$*,* g*, and* h*. From the term* $f(g,h)$*, three rewrite steps are possible. We have* $g > f$*,* $h > f$ *and* $g \mid h$*, and* $f \asymp f$*,* $g \asymp g$*, and* $h \asymp h$*.*

2. *It can happen that a HRS rewrite sequences is standard in the traditional sense, but its decomposition is not standard, as witnessed by the following example:*

$$\begin{array}{rcl} f(x.X(x)) & \rightarrow & g(X(h)) \\ g(Y) & \rightarrow & i(Y) \end{array}$$

The HRS rewrite sequence

$$f(x.j(x)) \rightarrow g(j(h)) \rightarrow i(j(h))$$

is standard, since the first step contributes to the second. However, its decomposition σ*,*

$$f(x.j(x)) \rightharpoonup (X.g(X(h)))(x.j(x)) \twoheadrightarrow_\beta g(j(h)) \rightharpoonup (Y.i(Y))(j(h)) \twoheadrightarrow_\beta i(j(h))$$

is not, since the second β-step caused by the first partial HRS-step is inside the second partial HRS-step, so

$$\sigma > f(x.j(x)) \rightharpoonup (X.g(X(h)))(x.j(x)) \to_\beta g((x.j(x))(h)) \rightharpoonup (Y.i(Y))((x.j(x))(h)) \twoheadrightarrow_\beta i(j(h))$$

Note that the resulting sequence is not the decomposition of a HRS rewrite sequence anymore.

3. *On the other hand, from each standardisation obtained by standardising the decomposition of a HRS-sequence we can obtain a HRS-sequence which is standard in the usual sense by projecting to β-normal form after each partial HRS-step. Note that this cannot increase the number of partial HRS-steps in the sequence.*

The $\Rrightarrow$ relation is the formalisation of the intuitive notion of 'removal of an anti-standard pair'. In a standard sequence first all the rewrite steps 'contributing' to the outermost symbol in the final term (possibly none) are performed, then the steps contributing to its daughters and so on. This holds recursively, i.e. the last rewrite step contributing to a symbol must 'produce' it so the other rewrite steps contributing to it must actually contribute to the left-hand side of the applied rule and are ordered in the same standard way, but now with respect to their contribution to that left-hand side. Up till now there's no ordering between the daughters. This will be provided later on, but first we show that $\Rrightarrow$ is strongly normalising and confluent.

In the case of sorting a list, termination of removal of 'inversions' follows easily from the fact that the length of the list is invariant under removal of inversions and a little acyclicity argument. In the case of standardisation the situation is not so clear since the length can both decrease and increase by removal of an anti-standard pair.

Theorem 2 (Standardisation). *1. $\Rrightarrow$ is strongly normalising and confluent modulo $||$[3].*

2. $\equiv\, = \,\Rrightarrow^! \,;\, || \,;\, {}^!\Lleftarrow$.

Proof

1. The idea to show strong normalisation is that all objects (i.e. sequences) obtained by $\Rrightarrow$ have bounded length, so by Königs Lemma there are only finitely many such and hence it suffices to show that $\Rrightarrow$ is acyclic. Here we prove acyclicity and the proof of boundedness is deferred to Appendix A.

   Let $\mathcal{U}$ and $\mathcal{V}$ denote the set of initial steps of all rewrites $||$-equivalent to σ and τ, their *visibly outermost* steps, and consider an arbitrary standardisation step $\sigma \Rrightarrow \tau$, that is, $\sigma \,||\, v \,;\, u \,;\, v \,;\, \phi > v \,;\, v' \,;\, \sigma' \,;\, \phi \,||\, \tau$ permuting $u \,;\, v$ into $v' \,;\, \sigma'$. We distinguish three cases:

   (a) if $u \in \mathcal{U}$,[4] then $\mathcal{V} \subseteq (\mathcal{U} - \{u\}) \cup \{v'\}$, because v' is outside u and σ' and the status of the steps in σ to the left or to the right of the permutation-step does not alter. For the former this is obvious, while for the latter it follows from the observation that $v \backsimeq v'$ and only these steps matter (not the steps in u or σ') for their status.

   (b) the case $v \in \mathcal{U}$ cannot occur, since the permutation-step is an outside-step (not a parallel-step).

   (c) in the remaining case no step in $v' \,;\, \sigma'$ contributes to $\mathcal{V}$, otherwise u would have been in $\mathcal{U}$. For the status of the other steps the same reasoning as in the first case pertains, so $\mathcal{V} = \mathcal{U}$.

   Hence, standardisation changes the set of visibly outermost steps iff the first case applies, and then either the cardinality is decreased or it remains the same and some step was replaced by a step outside it. Now, suppose for a contradiction that $\sigma \Rrightarrow^+ \sigma$ for some rewrite σ of minimal length. Each $\Rrightarrow$-step must be an instance of the third case, so in particular the initial step of σ is never part of a $\Rrightarrow$-redex and a shorter cyclic rewrite sequence is obtained by removing it from σ, contradicting minimality.

[3] see [Hue80] for the definition of confluence modulo.

[4] We confuse u with its unique origin along σ.

From the above, we know that every rewrite sequence can be further standardised to a standard one, so for a proof of confluence modulo $||$ it suffices to show that $\Rrightarrow$ has *unique normal forms modulo* $||$. We will in fact prove that standard rewrite sequences which are permutation equivalent are parallel, which is sound since $\Rrightarrow \cup || \subseteq \equiv$. This is shown in Appendix B.

2. In the case of orthogonal systems $\equiv$ is equal to the equivalence closure of $> \cup ||$, hence by definition of $\Rrightarrow$ also equal to the equivalence closure of $\Rrightarrow \cup ||$, yielding the result by the first item.

From the theorem it follows that one can effectively compute the standardisation of any rewrite sequence and also that permutation equivalence can be decided in an effective way.

The parallel relation $|$ on positions can be partitioned into $\langle|$ and $|\rangle$, where $u \langle| v$ if u is to the left of v. Formally, $\langle| =^{\text{def}} <_{lex} \cap |$ and $|\rangle =^{\text{def}} | \cap >_{lex}$ where $>_{lex}$ is the usual lexicographical ordering on positions. It is easy to see and show that the induced relation $|\rangle$ on rewrite sequences is strongly normalising and confluent, which means that uniqueness up to $||$ is sharpened into uniqueness proper by $|\rangle$-normalising standard rewrite sequences.

Remark 1. *In the case of* left-normal systems *(see e.g. [Klo80]) such as lambda calculus and Interaction Systems, this two phase process standardisation can be done in one go by normalising rewrite sequences with respect to* lexicographic *steps, i.e. permutation steps induced by $>_{lex}$, in a formula: $\Rrightarrow^! ; |\rangle^! = >^!_{lex}$. For non-left-normal systems only containment ($\supseteq$) holds in this formula as observed in [Hin78] for the case of lambda calculus with recursor. A simple TRS example showing this is:*

$$\begin{array}{rcl} f & \to & g \\ h(X,g) & \to & r \end{array}$$

where r is a term containing some occurrences of X. Consider the rewrite sequence: $\sigma\colon h(f,f) \to h(g,f) \to h(g,g) \to r^{[X \mapsto g]}$. The sequence σ is standard with respect to lexicographic standardisation, since there are no lexicographic anti-standard pairs. However, the first step does not contribute to the final step which takes place outside it, and indeed a $\Rrightarrow$-step can be performed:

$$\begin{array}{rl} \sigma & || \quad h(f,f) \to h(f,g) \to h(g,g) \to r^{[X \mapsto g]} \\ & > \quad h(f,f) \to h(f,g) \to g^{[X \mapsto f]} \twoheadrightarrow g^{[X \mapsto g]} \end{array}$$

5.2. Extraction

Extraction was described for Interaction Systems in [AL94] by adapting the method from lambda calculus [Lév78]. However a new operation (*shift*) had to be added to the lambda calculus extraction operation since in Interaction Systems redexes can be created (destroyed) and duplicated (erased) in the same step. A rule like $f(x.X(x)) \to X(g)$ might be applied to yield the step $f(x.h(x,x)) \to h(g,g)$ in which g is created and duplicated at the same time. This cannot happen in either TRSs or lambda calculus. Intuitively both occurrences of g have the same history but the lambda calculus extraction procedure does not show this. The shift operation was introduced in [AL94] exactly to correct this situation. In the example, the history of the right occurrence of g would be shifted towards that of the left occurrence, showing they're in the same backward family.

In our approach we decompose HRS steps into smaller components which either destruct and create symbols (partial HRS-steps) or duplicate and erase symbols (β-steps), but not both at the same time, circumventing the afore-mentioned problem. E.g. extracting the decomposition of the locked steps above yields

$$f(x.h(x,x)) \rightharpoonup (X.X(g))(x.h(x,x))$$

in either case Note that the resulting rewrite sequence is not the decomposition of a HRS rewrite sequence (as for standardisation).

Definition 13 (Extraction). *The extraction relation* $\triangleright$ *is the minimal relation on standard locked sequences, such that*

1. *It's compatible ([Bar84]), i.e. closed under* λ*-calculus contexts and substitutions,*
2. *Let* $\sigma: M \twoheadrightarrow N$ *be a rewrite sequence,* $u: N \rightarrow_{\phi} O$ *be a step* $\tau \triangleleft \partial: P \twoheadrightarrow Q$ *be a locked sequence.*
   (a) *If* u *is either a* β*-step, or a partial HRS-step,* $N \rightarrow_{\psi} O$ *such that* $O|_{\psi} = P$ *and* $\phi \mid \psi$ *then* $\sigma \,;\, u \,;_{\psi}\, \tau \triangleleft \partial \;\triangleright\; \sigma \,;_{\psi}\, \tau \triangleleft \partial$*, where the composition 'at position* ψ*' is defined in the obvious way.*
   (b) *If* u *is a partial HRS-step,* $N \rightarrow C[r(\vec{M}')] = O$ *and* $M'_m = \vec{x}.N$*, then* $\sigma \,;_m\, u \,;\, \tau \triangleleft \partial \;\triangleright\; \sigma \,;_m\, \tau \triangleleft \partial$*, where the sequences are confined to the* m^{th} *argument of the left-hand side and the right-hand side respectively.*
   (c) *If* u *is a* β*-step* $N = C[(x.P)(M')] \rightarrow_{\beta} C[P^{[x \mapsto M']}] = O$*, then* $\sigma \,;\, u \,;\, C[\tau \triangleleft \partial^{[x \mapsto M']}] \;\triangleright\; \sigma \,;\, C[(x.\tau \triangleleft \partial)(M')]$
   (d) *If* u *is a* β*-step* $N = C[(x.M')(P)] \rightarrow_{\beta} C[M'^{[x \mapsto P]}] = O$ *then* $\sigma \,;\, u \,;\, C[M'^{[x_m \mapsto \tau \triangleleft \partial]}] \;\triangleright\; \sigma \,;\, C[(x_m.M')(\tau \triangleleft \partial)]$*, where the locked sequence is confined to only the* m^{th}*-copy of* P *in both sequences.*

A sequence in extraction normal form is said to be extracted.

Extraction is clearly effective so combined with effectivity of standardisation this yields effectivity of the backward family relation to be defined next.

Definition 14 (Backward Family). *Define* $\succeq_{bw} =^{\mathrm{def}} \Rrightarrow^{!} \,;\, |\rangle^{!} \,;\, \triangleright^{!}$*. The* backward*-family relation is defined as* $\simeq_b =^{\mathrm{def}} \succeq_{bw} \,;\, {}_{bw}\preceq$*.*

It only remains to show that the backward family relation is an equivalence relation.

Theorem 3 (Extraction).

1. $\triangleright$ *preserves standardness.*
2. $\triangleright$ *is strongly normalising and confluent.*
3. $\simeq_b$ *is an equivalence relation.*

Proof

1. Since the transformations do not change the relative positions of the steps in either the initial sequence σ or the tailing sequence $\tau \triangleleft \partial$, the only change could be caused by some step w in τ which is 'outside' some step v in σ, and v did not contribute to w. Wlog we may assume v is the last step in σ and w the first step in τ. The statement follows by cases over the definition of extraction, each time deriving a contradiction with the standardness assumption.
2. Strong normalisation is trivial, so it remains to show confluence, or equivalently, uniqueness of normal forms. Consider two distinct locked sequences obtained by extraction from the same locked sequence. The steps in these sequences originate from unique steps in the original sequence. Consider the first step in the original sequence which has a 'descendant' in one but not in the other. So it *contributes* (see Definition 16) to one, but not to the other sequence. By the correspondence between the contribution relations for labelling and extraction, this would yield that their labellings differ, but this cannot be since labelling is preserved under extraction and both originate from the same (labelled) locked sequence.
3. Reflexivity and symmetry hold trivially. Transitivity follows from confluence and termination (modulo $||$ for $\Rrightarrow$) of $\Rrightarrow$, $|\rangle$, and $\triangleright$. $\Box$

6. Back and Forth

In this subsection we generalise the zigzag relation to orthogonal HRSs. This might be surprising since it has been claimed to be impossible [AL94] for its subclass of Interaction Systems. We feel that our generalisation follows quite naturally from our approach to rewriting.

The zigzag approach defines redex-occurrences to be in the same *back-and-forth* family if they can be related to each other via the equivalence closure of the *copy* relation (modulo *permutation equivalence*). Here, a redex v in the final term of a rewrite sequence $\sigma\,;\tau$ is said to be a copy of a redex u in the final term of σ, if it is a descendant of v along τ

Example 7 (See Example 3). *For the rewrite sequences we consider we don't need permutation equivalence.*

1. *Extending the partial HRS-step $\ddot{\sigma}$ ending in $(X.X(f))(x.h(x,x))$ by two β-steps yields the rewrite sequence $\tilde{\sigma}$ ending in $h(f,f)$, and this shows both occurrences of f in that term are copies of the redex f in the final term of $\ddot{\sigma}$. From this it follows that the occurrences are in the same back-and-forth family.*

2. *The rewrite sequence $\tilde{\tau}$ can be extended with the partial HRS-step $\breve{\tau}$: $g(\mu(x.f)) \rightharpoonup_{\epsilon} (X.f)(\mu(x.f))$ and the innermost f-redex in its final term is clearly a descendant of the redex f in its starting term.*

3. *The rewrite sequence $\tilde{\upsilon}$ can be extended with the rewrite sequence $\breve{\upsilon}$: $\mu(x.f) \rightharpoonup_{\mu} (X.X(\mu(x.X(x))))(x.f) \twoheadrightarrow_{\beta} (x.f)(\mu(x.f))$ and also here the innermost f-redex in the final term is a descendant of the redex f in the starting term. Since $\breve{\tau}$ and $\breve{\upsilon}$ end in the same term, we conclude from this and the previous item that the f redexes are in the same back-and-forth family.*

4. *The rule $f \rightarrow_f f$ does not do any copying (only destruction and creation), and it follows directly from the definition of back-and-forth families that the initial and final terms of $f \rightarrow_f f$ cannot be related to each other; they're not in the same back-and-forth family.*

If HRS steps are considered (not decomposed ones), it is not possible to find a suitable zigzag in the third item as was noted in [AL94, Exa. 10], due to the fact that completing the $\breve{\tau}$ and $\breve{\upsilon}$ rewrite sequences to (decompositions of) HRS steps (by β-normalising their final terms to f) leads to HRS steps which 'create and erase', in particular the innermost occurrence of f witnessing the equivalence is erased.

Definition 15. *Zigzag*

1. *Let $M \rightarrow_{\phi} N$ be either a partial HRS-step or a β-step. A part ∂ of N is said to be a* copy *of the same part in M, if*

   (a) *No position of ∂ is inside ϕ and then it's a copy of itself in M,*

   (b) *∂ is inside an argument of the partial HRS-step at ϕ. This means that it uniquely stems from a part of (an instance of) the corresponding bound hole in M,*

   (c) *∂ is completely inside the (substitution instance of) the body of the β-step at ϕ. This means that it descends from a unique part in the body of the step in M.*

   (d) *∂ is completely inside an occurrence of the argument of the β-step at ϕ. This means that it descends from a unique part in the argument of the step in M.*

2. *The* back-and-forth *family relation $\simeq_z$ is defined as the equivalence closure of the copy and the permutation relations.*

Note that the four cases in the definition of the copy relation correspond to the four cases in the definition of extraction. The first case however is somewhat extended, since here we do not assume standardness of the locked sequences. By definition $\simeq_z$ is an equivalence relation.

7. Back = Forth = Back and Forth

Our main results show that it is not a coincidence that for the examples the three distinct family relations coincide. They are shown to be equivalent for all OHRSs. This requires quite some technicalities but is in principle an adaptation of the techniques in the appendices of [AL94].

The easy part is to show that if two locked sequences extract into the same sequence, labelling them yields identically labelled locks.

Proposition 2. *Labelling is preserved by* $\Rrightarrow$, $|\rangle$, *and* $\triangleright$.

Proof For the first two, this follows from the fact that labelling an elementary diagram yields a labelled elementary diagram. This is a tedious excercise in orthogonality. Preservation of $\triangleright$ is a consequence of Proposition 3.

We define two *contribution* relations, one via labelling and one via extraction and show them to coincide.

Definition 16 (Contribution). *Let* $u\,;\sigma\lhd\partial$ *be a locked standard sequence.*

1. *The step* u *is said to* forwardly contribute *to* ∂ *if its degree is a* sublabel *of the degree of the lock in the initial labelling (where sublabel is defined in the obvious way).*

2. *The step* u *is said to* backwardly contribute *to* ∂ *if the sequence extracts into* $u\,;\sigma'\lhd\partial$ *for some* σ'.

Proposition 3. *Forward contribution is backward contribution.*

Proof This proposition corresponds to [AL94, Pro. 6] and is proven in the same way by induction over the length of the locked sequence.

Theorem 4. *Forward families are the same as backward families.*

Proof This is [AL94, Thm. 6]. One half is easy and follows directly from Proposition 2. The other half is difficult and makes essential use of the notion of *connection* to be given below. One proves that if two locked sequences do not extract to the same sequence, their locks are differently labelled after inital labelling. This follows essentially the proof in op. cit. The key part is to show that if both sequences start with the same initial step u, then from the induction hypothesis for the tailing sequences, the result for the sequences themselves follows. However, the induction hypothesis applies for initial labellings of the tailing sequences and if u is a *duplicating* β*-steps* $C[(x.M)(N)] \to_\beta C[M^{[x\mapsto N]}]$ then (and only then) initiality of the labelling of its results does not hold true anymore (elements in N have the same label in all copies). However, one can still 'see' in which of the duplicated parts steps take place via the connection of these elements to the (label of its) x's for which it was substituted (all x have distinct labels by initiality). Connectivity is defined in such a way that this 'seeing' possibility is preserved under rewriting. Cf. [AL94, App. A] for all the details. □

Definition 17 (Connection). connectivity *of labels is defined as follows.*

1. *One can decompose every edge label into labels which are not concatenated labels. Each of these labels is* connected *to its neighbour(s) and the ones next to a node are also connected to that node.*

2. *An 'interfacenumber'* m *next to the composed label* $\langle\vec{\alpha}, l \to r, n\rangle$ *is connected (via a jump) to the corresponding element in the labelled left-hand side determined by the composed label.*

3. *Connection is closed under making it part of a composed label, (i.e. if a left-hand side is put into a label in a partial HRS step, connectivity inside that label is the same as it was for the left-hand side).*

4. An element inside a composed label is connected to that same element of its identical neighbour.

Only the first item is needed in the case of λ-calculus since there, no elements can be created by rewrite steps. The other three items are meant to deal with HRS-steps. The first to have after performing a HRS step, still access to connectivity as before the step. The second to make connectivity 'closed under contexts' and the third to allow to move from one element of the right-hand side to another.

Since the definitions of extraction and zigzag make use of similar concepts, it's no surprise that, like for λ-calculus, they can easily be proven to be equal.

Proposition 4. *Backward families are the same as back-and-forth families.*

Proof Assume two locked sequences extract into the same sequence after standardisation. The four extraction steps can be abstractly depicted as

$$\begin{tikzcd} & O \arrow[r, two heads, "\tau"'] & O'\lhd\partial \\ M \arrow[r, two heads, "\sigma"'] & N \arrow[u, "u"'] \arrow[r, two heads, "\tilde{\tau}"'] \arrow[ur, phantom, "\bigtriangledown"] & N'\lhd\partial \end{tikzcd}$$

transforming the top sequence $\sigma\,;u\,;\tau$ into the bottom one $\sigma\,;\tilde{\tau}$.

1. If u is not a β-step, or τ is inside the body of the β-step, we have $N'\lhd\partial \to_{u'} O'\lhd\partial$ for a unique descendant u' of u.

2. If τ is confined to one copy of the argument of the β-step u, then we can define the *parallelisation* $\bar{\tau}$ of τ by performing the each step of τ on each of the copies, ending in a term $\bar{O}'\lhd\partial$. Since the parallel steps are all disjoint from ∂ in $O'\lhd\partial$, the lock of $\bar{\tau}$ is a copy of that of τ. Relating $\tilde{\tau}$ to $\bar{\tau}$ is performed as in the previous item.

We conclude that the top and bottom sequences can always be related to each other via a zigzag. The opposite direction is just as easy. $\Box$

8. Further Research

Once a solid notion of family is established, the next question is whether it is possible to implement an evaluator for orthogonal HRSs in which redexes in the same family are actually represented by the same structure, and which only ever contracts redexes which will contribute to the final result of the computation.

For the second, a generalisation to the higher-order case of the theory of *sequentiality* and *neededness* as founded in [HLa, HLb] for orthogonal TRSs, is required. The theory of neededness has already been generalised to the second-order framework of Expression Reduction Systems (ERSs [Kha]) and we expect no difficulties in extending those results to HRSs. Concerning sequentiality, we expect that the methods and results of [KM91] for *strong sequentiality* extend directly to HRSs.

For the first, we expect to achieve optimality for all orthogonal HRSs, showing that this is possible due to the logical nature of substitution, not rules (cf. [Lan93]).

References

[AL94] Andrea Asperti and Cosimo Laneve. Interaction systems I: The theory of optimal reductions. *Mathematical Structures in Computer Science*, 4:457–504, 1994.

[AL96] Andrea Asperti and Cosimo Laneve. Interaction systems II: The practice of optimal reductions. *Theoretical Computer Science*, July 1996. To appear.

[Bar84] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, revised edition, 1984.

[BBKV76] H. P. Barendregt, J. Bergstra, J. W. Klop, and H. Volken. Degrees, reductions and representability in the λ-calculus. Report 22, Rijksuniversiteit Utrecht, February 1976. The Blue Preprint.

[BG93] M. Bezem and J. F. Groote, editors. *Proceedings of the International Conference on Typed Lambda Calculi and Applications, TLCA'93, March 1993, Utrecht, The Netherlands*, volume 664 of *Lecture Notes in Computer Science*, Berlin Heidelberg, 1993. Springer-Verlag.

[Dan90] Vincent Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du λ-calcul)*. Thèse de doctorat, Université Paris VII, 20 Juin 1990.

[DJ] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In [Lee90, Ch. 6].

[GLM] Georges Gonthier, Jean-Jacques Lévy, and Paul-André Melliès. An abstract standardisation theorem. In [LIC92, pp. 72–81].

[Gro] P. de Groote. The conservation theorem revisited. In [BG93, pp. 163–178].

[Hin78] R. Hindley. Standard and normal reductions. *Transactions of the American Mathematical Society*, 241:253–271, July 1978.

[HLa] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, I. in [LP91, Ch. 11].

[HLb] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, II. in [LP91, Ch. 12].

[HP95] Michael Hanus and Christian Prehofer. Higher-order narrowing with definitional trees, September 1995. Draft.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery*, 27(4):797–821, October 1980.

[Kha] Zurab Khasidashvily. On higher order recursive program schemes. In [Tis94, pp. 172–186].

[Klo80] J. W. Klop. *Combinatory Reduction Systems*. PhD thesis, Rijksuniversiteit Utrecht, June 1980. Mathematical Centre Tracts 127.

[KM91] Jan Willem Klop and Aart Middeldorp. Sequentiality in orthogonal term rewriting systems. *Journal of Symbolic Computation*, 12(2):161–195, August 1991.

[KW94] A. J. Kfoury and J. B. Wells. New notions of reduction and non-semantic proofs of β-strong normalization in typed λ-calculi. Technical Report 94–014, Boston University Computer Science Department, 1994. (See [KW95]).

[KW95] A. J. Kfoury and J. B. Wells. Addendum to "new notions of reduction and non-semantic proofs of β-strong normalization in typed λ-calculi". Technical Report 95–007, Boston University Computer Science Department, March 31 1995.

[Lan93] Cosimo Laneve. *Optimality and Concurrency in Interaction Systems*. PhD thesis, Dipartimento di Informatica, Università degli Studi di Pisa, March 1993.

[Lee90] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volume B : Formal Models and Semantics. Elsevier Science Publishers B.V., Amsterdam, 1990.

[Lév78] Jean-Jacques Lévy. *Réductions correctes et optimales dans le λ-calcul*. Thèse de doctorat d'etat, Université Paris VII, ?? 1978.

[LIC91] *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science, Amsterdam, The Netherlands, July 15–18, 1991*, Los Alamitos, California, 1991. IEEE Computer Society Press.

[LIC92] *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California, June 22–25, 1992*, Los Alamitos, California, 1992. IEEE Computer Society Press.

[LP91] Jean-Louis Lassez and Gordon Plotkin, editors. *Computational Logic: Essays in Honor of Alan Robinson*. The MIT Press, Cambridge, Massachusetts, 1991.

[Mar92] Luc Maranget. *La stratégie paresseuse*. Thèse de doctorat, Université Paris VII, 6 juilliet 1992.

[Mel95] Paul-André Melliès. *Description Abstraite des Systèmes de Réécriture*. Thèse de doctorat, Université Paris VII, Draft of September 28, 1995.

[Mil] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In [SH91, pp. 253–281].

[Ned73] R. P. Nederpelt. *Strong Normalization in a Typed Lambda Calculus with Lambda Structured Types.* PhD thesis, Technische Hogeschool Eindhoven, June 1973.

[Nipa] Tobias Nipkow. Higher-order critical pairs. in [LIC91, pp. 342–349].

[Nipb] Tobias Nipkow. Orthogonal higher-order rewrite systems are confluent. in [BG93, pp. 306–317].

[NM94] A. Nerode and Yu. V. Matiyasevich, editors. *Logical Foundations of Computer Science, Third International Symposium, LFCS'94, St. Petersburg, Russia, July 1994, Proceedings*, volume 813 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Oos94] Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, March 1994.

[OR] Vincent van Oostrom and Femke van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In [NM94, pp. 379–392]. Full version available as Technical Report, CS-R9501, CWI, January 1995.

[SH91] P. Schroeder-Heister, editor. *Extensions of Logic Programming, International Workshop, Tübingen, FRG, December 8–10, 1989*, volume 475 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1991.

[Tis94] Sophie Tison, editor. *Trees in Algebra and Programming – CAAP '94, 19th International Colloquium, Edinburgh, U.K., April 11–13, 1994, Proceedings*, volume 787 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Wol93] D. A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1993.

A. Boundedness of Standardisation

In this appendix we present the proof that all rewrite sequences obtained by standardisation are bounded in length by the initial rewrite sequence.[5] This was suggested to hold for orthogonal CRSs in [Klo80, Rem. II.6.2.8.10] and we will prove it to be true for orthogonal HRSs. This will require some effort since the method described for CRSs was based on [Klo80, Thm. II.6.2.4] stating that every orthogonal CRS having a 'decreasing redex labelling' is strongly normalising, but this result is false as observed in [Mel95, Sec. 6.2.2].

Definition 18 (Non-erasing HRS). *1. The β-rule is split into two parts.*

$$\begin{array}{lll}(x.M)(N) & \to_{\beta_I} & M^{[x\mapsto N]} \text{ if } x \in M \\ (x.M)(N) & \to_{\beta_K} & \langle M, N\rangle \text{ if } x \notin M\end{array}$$

Doing this, would reduce the rewrite power since a memory*-term $\langle N, M\rangle$ can 'get in the way' of both β_I-redexes and partial HRS-steps. To this end, we employ so-called* shift *rules, which shift memory towards the root in such situations.*

$$\begin{array}{rll}\langle (y.M), N\rangle(O) & \to_{shift_\beta} & \langle (y.M)(O), N\rangle \\ l' & \to_{shirt_\mathcal{H}} & \langle l'', \vec{X}\rangle\end{array}$$

In the shift$_\mathcal{H}$-rule, l' is obtained from an arbitrary left-hand side of a rule, by replacing some subterms (except for subterms of bound holes) in it by memory $\langle Y, X\rangle$ (variables each time fresh). In l'' this memory is replaced by Y, while X is shifted towards the root (into $\vec{X}$, taking care to properly bind variables occurring in X). By fixing an order for this shifting, the rule is made deterministic.

2. The resulting non-erasing *HRS is denoted by $\mathcal{H}^{ne}$.*

[5] Note that the length of rewrites in a permutation equivalence class is in general unbounded due to erasing steps.

Note that the shift_β rule has neither overlap with itself nor with the β_I rule. Furthermore, overlap between $\text{shift}_\mathcal{H}$-rules themselves or with $\mathcal{H}$-rules would imply overlap between $\mathcal{H}$-rules which is not possible by the orthogonality condition on $\mathcal{H}$.

A HRS and its non-erasing version correspond closely to another.

Theorem 5. *If $\mathcal{H}$ is orthogonal, then so is $\mathcal{H}^{ne}$ and*

1. *if $M \to_{\mathcal{H}^{ne}} N$ for either a partial HRS-step or a β-step then $\lfloor M \rfloor \to_{\mathcal{H}}^{=} \lfloor N \rfloor$, where $\lfloor M \rfloor$ denotes the forget-normal form of M, obtained by projecting memory onto its first component.*

2. *If $\lfloor M \rfloor \to_{\mathcal{H}} N'$, for either a partial HRS-step or a β-step, then $M \twoheadrightarrow_{shift} O \to_{\mathcal{H}^{ne}} N$ ending in the corresponding (non-erasing) step, such that $\lfloor N \rfloor = N'$.*

Proof ortho

1. Observe that adding the forget-rule to β_I and β_K yields the same equivalence relation as β, since the β_K-normal forms of the left- and right-hand sides of shift rules are the same. Then,

$$\begin{tikzcd}
C\boxed{l} \arrow[r, "\aleph"'] \arrow[d, "\beta_{IK}"] & C\boxed{r} \arrow[d, "\beta_{IK}"'] \\
M \arrow[r, "\mathcal{H}^{ne}"'] \arrow[d, "forget"] & N \arrow[d, "forget"] \\
\lfloor M \rfloor \arrow[r, "\mathcal{H}"', "\circ" marking] & \lfloor N \rfloor
\end{tikzcd}$$

   and we conclude by noting that the development step must be either an empty or a single step depending on whether it was erased or not.

2. In the non-erasing version, the memory might be in the way, but the shift-rules are exactly there to resolve this problem. After shifting the step can be simulated. □

In the following abstract rewriting lemma taken from [Klo80, Cor. I.5.19], *increasingness* expresses that there is a strictly increasing measure m from rewrite sequences to natural numbers.

Lemma 1. *A locally confluent, increasing, weakly normalising abstract rewriting system is strongly normalising.*

Proof Consider for a contradiction a term M allowing an infinite rewrite $M \to N \to^\omega \ldots$ as well as a rewrite $\sigma: M \to^! Q$ for which $m(Q) \dot{-} m(M)$ is minimal. If M exists, then $M \to O \to^! Q$ for some O. By local confluence $N \twoheadrightarrow P \twoheadleftarrow O$ for some P. By the minimality assumption and increasingness O is strongly normalising together with local confluence yielding confluence of O. Hence $O \twoheadrightarrow P \to^! Q$ and $N \twoheadrightarrow P \to^! Q$ contradicting minimality of σ since $m(N) > m(M)$ by increasingness. □

Note that increasingness with respect to a well-founded order instead of the natural order on natural numbers will not do. One can however sharpen the lemma somewhat by requiring the rewriting system to be only *eventually* increasing, that is, requiring $\to^+$ to be well-founded on $\{N.M \twoheadrightarrow N \,\&\, m(M) = m(N)\}$ for each M.

The 'memory'-method is essentially based on the possibility of defining a strictly increasing measure for all rewrite rules. However some 'memory control rules' found in the literature do not satisfy the strictness requirement and hence some proofs are incorrect (cf. the shift-rule in [Klo80, Pro. II.4.9] the β_S-rule in [Gro, Thm. 27]). The liberalisation to eventual increasingness is a possible solution, since these 'memory control rules' on their own are strongly normalising.

Definition 19 (Labelled non-erasing HRS). *Labelling for a non-erasing is defined as for ordinary HRSs.*

Lemma 2 (Bounded Labelling Lemma). *Every rewrite sequence in a labelled non-erasing HRS for which the size of a label is bounded is finite.*

Proof[Sketch] We check the conditions of (the sharpened version of) Lemma 1. Take as measure the length of all the labels in a term.

1. First remark that in every non-shift step, at least one label gets larger and no labels are erased so these steps are increasing. In a shift-step the measure doesn't change, but these rules are strongly normalising.
2. Local confluence of the labelled non-erasing HRS follows from local confluence of the non-erasing HRS, which follows from orthogonality.
3. A normal form from the initial term is obtained as follows, contract in each step all partial HRS- and β-steps. Repeat this until no steps are possible. (this stops since each created redex has a longer label than the redex by which it was created.) □

Since labelling a finite rewrite sequence yields finite labels in its final term, we have the following theorem.

Theorem 6 (Finite Families). *Every rewrite sequence in which redexes from a finite number of families are contracted is finite.*

The theorem finishes the proof of Theorem 2 since the initial number of families contracted is finite (the length of the rewrite is finite) and this number does not increase by standardisation.

Remark 2. *The proof of the Finite Families Theorem is usually based on a 'decreasing labelling' method which does not depend on orthogonality of the rewrite system (cf. [Lév78, Thm. II.4.4.6], [Mar92, Prop. 3.3.11], [Mel95, Thm. 6.5]). We have employed and corrected the method introduced in [Klo80, Thm. II.6.2.4] which only applies to orthogonal systems.*

Memory History The idea of 'memory' was introduced as early as 1973 by Nederpelt ([Ned73], under the name of 'scar') to reduce strong normalisation of simply typed lambda calculus to weak normalisation of a transformed calculus. A similar method was employed in [Klo80] showing strong normalisation of lambda calculus with Hyland-Wadsworth labels (implying that of simply typed lambda-calculus) and used as a tool to prove properties of orthogonal CRSs. The idea of our method is essentially the same, but in Klop's method memory is added to the (CRS) rewrite rules, while we add memory to the substitution calculus. We think this is not only better from our conceptual point of view ('erasure is part of the substitution calculus'), but also that it indicates exactly why his Theorem II.6.2.4 fails (see [Mel95, Sec. 6.2.2]): although adding memory to rewrite steps prevents erasure of existing redexes, it does not prevent erasure of redexes created by a rewrite step, i.e. creation and erasure in the same rewrite step.

In [Gro, KW94] a related, but different idea for proving strong normalisation of (typed) lambda calculi is employed. The difference is that in their methods steps which might have an erasing effect (K-redexes) are not performed at all, instead of being performed in a lazy way as in the previous methods. The problem to be solved then is that not-performing K-redexes might prevent creation of (non-erasing) I-redexes. A solution is to move the K-redexes out of the way when harmful. This is an instance of the general idea of making 'visible' redexes explicit, for which the following two rules were presented in [Dan90, Déf. 1–3] (using his notation):

$$
\begin{array}{rcl}
((\lambda x a)b)c & \rightarrow^{op} & (\lambda x(a)c)b \text{ if } x \notin c \\
((\lambda x \lambda y a)b & \rightarrow^{op} & \lambda y(\lambda x a)b \text{ if } y \notin b
\end{array}
$$

The first rule corresponds to the β_S-rule in [Gro], whilst the second rule is the γ-rule in [KW94]. Defining $\langle M,N\rangle_x =^{\mathrm{def}} x.M(N)$ these rules read in our notation (with similar side-conditions):

$$
\begin{array}{rcl}
\langle M,N\rangle_x(O) & \rightarrow_{shift\_app} & \langle M(O),N\rangle_x \\
\langle y.M,N\rangle_x & \rightarrow_{shift\_abs} & y.\langle M,N\rangle_x
\end{array}
$$

The first rule can be viewed upon as shifting the memory outside an application (towards the root), while the second rule can be thought of as shifting the memory inside an abstraction (towards the leafs) and indeed as one easily sees either of them suffices to prevent blocking I-redexes by K-redexes.

We prefer the lazy-erasure scheme above the non-performance scheme since the former is a refinement of the β-rule (allowing for a smooth refinement of labelling as well), while the latter seems rather ad hoc. Moreover we observe that rewriting with the 'memory operator' stands to pessimal rewriting in the same way as rewriting with 'control operators' stands to optimal rewriting and the techniques employed in both methods are the same.

A.1. Pessimality

Dually to optimality, given a family rewrite σ one can search for a permutation equivalent one having the highest cost, i.e. a pessimal rewrite. Formally, pessimality is defined as optimality with respect to $\geqslant$ instead of $\leqslant$.

If no optimal rewrite exists it means that σ is finite and there exists an infinite rewrite $\tau \equiv \sigma$. The Finite Families Theorem shows that this infinite behaviour must take place in a nonneeded part, i.e. a part which was erased by σ.

The proofidea was to show that erasure can always be delayed, without loosing termination. The practice of delaying erasure is analogous to the practice of delaying duplication, as found in the study of optimality (see e.g. [AL96]). Whereas in the study of optimality duplication is delayed by means of control nodes (fans, brackets and croissants), in the study of pessimality erasure is delayed by means of the memory construct.

B. Standardisation has unique normal forms modulo $||$

Proof So suppose $\sigma \equiv \tau$ are standard, but not $\sigma \mid\mid \tau$. We consider the sets $\mathcal{U}$, $\mathcal{V}$ of visibly outermost steps of σ, τ and distinguish two cases:

1. if $\mathcal{U} = \mathcal{V}$, we repeat the process now with $\mathcal{U}\backslash\sigma$ and $\mathcal{V}\backslash\tau$ ($\mathcal{U}\backslash\sigma$, σ *after* $\mathcal{U}$ is the usual orthogonal projection of σ over $\mathcal{U}$ (cf. [HLa, Klo80]) and observe that these are again permutation equivalent standard rewrite sequences). This case does not apply indefinitely since the length of the rewrite sequences decreases and the initial sequences are not parallel by assumption.

2. otherwise, let u be (wlog) in $\mathcal{U} - \mathcal{V}$ so $\sigma \mid\mid (u\,;\sigma')$. Consider the construction of $u\backslash\tau$. If τ is the empty sequence, then we have a contradiction with $\sigma \equiv \tau$. Otherwise $\tau =^{\mathrm{def}} v\,;\tau'$, and we distinguish cases according to the relationship between u and v.

   (a) if $u \mid\mid v$ then $v\backslash u = \{u'\}$, for some u' and we repeat the process for u' and τ',

   (b) if $u < v$, then the same holds as in the previous case, but we also remember this last inside-step,

   (c) if $u \gtrless v$, then if there does not exist a last inside-step, only the first case has been applied and we have a contradiction with the assumption that $u \in \mathcal{U} - \mathcal{V}$. On the other hand if there does exist a last inside-step w, this contradicts standardness of τ since we have a $\Rrightarrow$-redex by ($||$)-preponing v until w.

   (d) if $u > v$, we again make a case distinction. Let the original rewrite be $\tau =^{\mathrm{def}} \tau'\,;v\,;\tau''$. Note that every step in τ' is parallel to or inside u and hence also v. If σ is the empty sequence (initially false), then we have a contradiction with $\sigma \equiv \tau$. Otherwise $\sigma =^{\mathrm{def}} u\,;\sigma'$, and we distinguish cases according to the relationship between v and u.

      i. if $v \mid\mid u$, then also $(\tau'\backslash u) \mid\mid v$, and we repeat the process for σ' and $u\backslash\tau$ (note that v has a unique descendant and that the descendants of τ' are all parallel to or inside it),

      ii. if $v < u$, then the same holds as in the previous case, but we remember u.

      iii. if $v \gtrless u$ or $v > u$, then there does exist a last $<$-step w which contradicts standardness of σ since we have a $\Rrightarrow$-redex by ($||$-)postponing w until u. $\square$

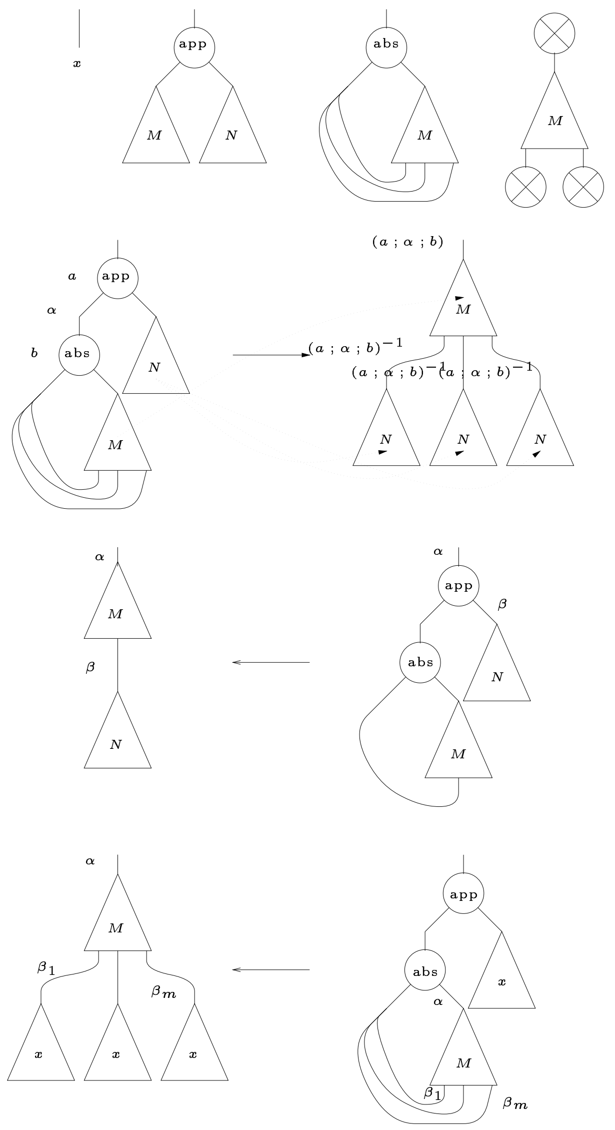

Figure 1: (Labelled) Bourbaki-representation

The top row presents the graphical representation of λ-terms where the root and the free variables are connected to 'terminator' nodes. The next row shows a labelled β-reduction step, where $\cdot^{-1}$ denotes string inverse. while the last two rows show the β-expansion steps needed. The first way is needed for splitting off argument parts (N) from the redex part (M) in a partial HRS-step. The second way of labelled β-expansion is needed to split bindings (of x) between redex parts and argument parts.

$$\begin{tikzcd} & M \arrow[dl, "u"'] \arrow[dr, "v'"] & \\ N \arrow[dr, "v"'] & \| & O \arrow[dl, "u'"] \\ & P & \end{tikzcd} \qquad \begin{tikzcd} & M \arrow[dl, "u"'] \arrow[dr, "v'"] & \\ N \arrow[dr, "v"'] & > & O \arrow[dl, "\mathcal{U}'"] \\ & P & \end{tikzcd} \qquad \begin{tikzcd} & M \arrow[dl, "u"'] \arrow[dr, "u"] & \\ N & \bumpeq & N \end{tikzcd}$$

Figure 2: Parallel, Outside and (orthogonal) Overlap